# From Steady-State Probabilities to Performance Measures for Complex Reactive Systems specified in Statecharts

Nandamudi L. Vijaykumar[1], Sóstenes Pereira Gomes[2] , Marcelino Silva da Silva[3], Solon Venâncio de Carvalho[1], Carlos Renato Lisboa Francês[3]

[1]Laboratory of Computing and Applied Mathematics (LAC)
[2]Graduate Student, Applied Computing (CAP)
National Institute for Space Research
P. O. Box 515
12245-970 São José dos Campos, SP, Brazil
{vijay, sostenes, solon}@lac.inpe.br

[3]Department of Electrical Engineering
Federal University of Pará (UFPA), Belém, PA, Brazil
{marcelino, rfrances}@ufpa.br

**Abstract**

Performance evaluation of modern complex systems has become compulsory in order to understand a system's behavior before it is constructed or implemented. Usually either Simulation of Analytical approaches are used to obtain such understanding. However, the system must, somehow, be represented. The paper describes PerformCharts, a tool to analytically obtain performance evaluation of a system specified in Statecharts. Until recently, the information obtained were just steady-state probabilities. Therefore, this paper discusses how such probabilities were used in order to obtain the performance measures of interest.

## 1. Introduction

Several applications face with the necessity in evaluating the performance of a given system. By evaluating the performance, one can determine how reliable the system is besides finding out many other parameters of interest. The systems where their evaluation is studied and analyzed are known as performance models.

Performance models are, in general, stochastic models. In these models, system behavior is usually modeled as a discrete event driven system and represented by a state-transition diagram. In this diagram each state represents a physical state of several components of the system (idle, busy, failure, etc.) and the transitions among the states occur through events that correspond, for example, to some perturbation in the system such as a failure.

Due to their strong mathematical basis, Continuous-Time Markov models are frequently used to model reliability and system performance. In these models, reliability and many performance measures can be obtained and these are probability functions of occupying the states during a certain period of time or in a long run horizon. Markov models can be represented by state-transition diagrams. However, these diagrams make the representation of the behavior of the modeled complex systems complicated. For example, by covering all the combinations of a concurrent model the diagram may increase exponentially and consequently leads to a state blow-up phenomenon (Drusinsky, 1994). A detailed representation may lead to hundreds of thousands of states and notions of concurrency and interdependence among the system components become difficult to handle. This difficulty brings up the necessity of investigating the use of high level tools for specifying complex systems from which a mathematical model can be automatically generated.

Complex systems involving parallelism, synchronization and interdependence of subsystems have become popularly known as ***reactive systems***. The main characteristic of a reactive system is its whole behavior is based on reaction to stimulus, also known as events, received by internal and external media (Harel, 1987). These complex systems are heavily based on controls or

events. In these cases data is not handled just by the description of to and from activities. The reaction to different kinds of events, signals and conditions does take place in complex ways. In fact, a vast majority of these complex systems in day to day applications are rather reactive in nature.

The objective of this paper is to show that a tool, PerformCharts, has been developed in which a reactive system is specified in Statecharts and a corresponding Markov chain is obtained. With the Markov chain at hand, by applying a numerical method to solve the chain, steady state probabilities are easily obtained. An important issue missing in this tool are the performance measures that are functions of the obtained probabilities. Therefore, the paper also discusses the inclusion of such calculations within the tool. A case study is presented in order to show the functionality of the tool. The paper is organized as follows: Section 2 discusses very briefly PerformCharts as well as a very short description of Statecharts. Section 3 shows the inclusion of the implementation to determine the performance measures. Section 4 covers a case study while Section 5 concludes the paper with some remarks.

## 2. PerformCharts: Obtaining Markov Chain from Statecharts

Before formally presenting PerformCharts tool, some essential elements of Statecharts are briefly described. Statecharts have a graphical language to specify reactive systems. They have been originally developed to represent and simulate real time systems. They have an added value of being formal (Harel et al, 1987) and (Harel & Politi, 1998) and their visual appeal along with the potential features enable considering complex logic to represent the behavior of reactive systems. They originated from state-transition diagrams and these diagrams have been extended with notions of hierarchy (depth), orthogonality (parallel activities) and interdependence (broadcast-communication). Statecharts depend on the following elements in order to represent a reactive system: states, events, conditions, actions and transitions. It is also possible to define variables and expressions. Events are divided into two categories: external and internal or immediate. External are those that have to be explicitly stimulated whereas internal are those that are automatically sensed by Statecharts dynamics and reaction takes place. This is the same as defined in Statecharts. However, for the application in performance evaluation, the external events carry a stochastic information which is exponentially distributed in order to be associated with Markov chains. Actions, within the context of performance evaluation, are considered as internal events that affect other orthogonal components.

States are clustered to represent depth, thus enabling to combine a set of states with common transitions into a super-state. Super-states are usually refined into further sub-states in a top down approach. State refinement can be achieved by XOR and AND decompositions. The former decomposition is employed whenever an encapsulation is a must to improve the clarity of the visualization. When an XOR super-state is active, one (and only one) of its sub-states is indeed active. The latter approach, AND decomposition, is used to represent concurrency and when active, all of its sub-states are active. The state that contains no more further refinements is known as BASIC.

In Statecharts the global state of a given model is referred to as a configuration that is the active basic states of each orthogonal component. In Statecharts, by definition, when modeling a given system, there must always be an initial state also known as default state. This is the entry point of the system. Another way to enter a system is through its history, i.e. when a system is entered the state most recently visited is activated, thus bypassing the initial state. In order to indicate that history is to be used instead of entry by default, the symbol H is provided. It is also possible to use the history all the way down to the lowest level as defined in the Statecharts formalism (Harel, 1987) by applying the symbol H*.

This was just a very short introduction to Statecharts. Now, the description on PerformCharts is in order. A Markov Chain, within the scope of this work, a Continuous-Time Markov Chain, consisting of transition rates among states is the input to the available numerical methods

(Philippe et al., 1992) and (Silva & Muntz, 1992) to determine the steady-state probabilities. Therefore, the problem of constructing a Markov model will be solved if the model represented in Statecharts generates the Markov chain that corresponds to the behavior of the specified model.

Algorithm, in this paper, is explained in an informal manner. Formal algorithms are given in (Vijaykumar et al, 2006). Once the model is specified in Statecharts, the first step is to check which events are to be triggered for the initial configuration determined by default states of each parallel component. Internal (or immediate) events are the ones that are sensed and automatically triggered. As long as such events are active for the initial and resulting configurations, reactions continuously take place by changing one configuration to another until a configuration is reached from which no more internal events are enabled. The next step deals with the stochastic events which are explicitly stimulated. Therefore, for the resulting configuration from internal events, enabled stochastic events are listed and triggered so that transitions are fired yielding new configurations. In order to make the association of a Statecharts model with a Markov Chain, the only type of events considered are those that follow a stochastic distribution. In particular, for Continuous-Time Markov Chains, this distribution has to be exponential. Once a configuration is obtained, internal events, if enabled, are triggered, firing transitions to yield new configurations. In both the cases, actions are also considered if they are associated in a transition. In this case a reaction occurs whenever appropriate, based on the action which is considered as an internal event. This process continues until all the configurations have been expanded. The result is a structure with a source configuration, stimulated stochastic event (along with its rate), and the target configuration. This structure is, indeed, a Markov Chain (specified as a transition matrix) with which steady-state probabilities can be determined.

Now, to understand the whole process, from specification to generation of the Markov Chain, consider a Standby Redundancy System with two machines (Machine 1 and Machine 2) where each machine is a standby for the other (Viswanadham & Narahari, 1988). Machine 1 has a priority to process a job when both the machines are idle and in working conditions. Machine 2 takes up the job whenever Machine 1 fails and simultaneously, Machine 1 is repaired. Yet Machine 2 may also break down while it is processing. In this case the job will be transferred to Machine 1 if it is in working condition. This model is shown in Figure 1.

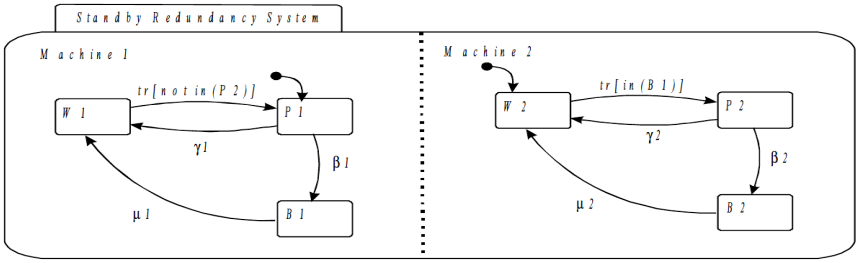


Figure 1. Standby Redundancy System specified in Statecharts

Figure 1 shows two parallel components Machine 1 and Machine 2 whose sub-states are *Waiting* (W1 and W2), *Processing (*P1 and P2) and *Failure (*B1 and B2). The initial states for Machine 1 and Machine 2 are respectively P1 and W2. Events $\gamma_1$ and $\gamma_2$ refer to end of processing. A failure occurs through $\beta_1$ and $\beta_2$ whereas end of repair through $\mu_1$ and $\mu_2$. Machine 1 has a priority over Machine 2 which takes over only when Machine 1 breaks and this fact uses the true conditioned event (immediate event) *tr[in(B1)]*. Now, this representation is converted into a Markov chain, shown in Figure 2.

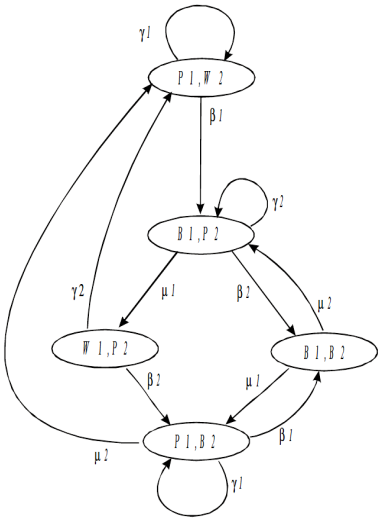


Figure 2. Markov chain of Figure 1

This state-transition diagram has a one-to-one correspondence with a transition matrix where rows and columns correspond to the configurations and the elements of the matrix correspond to the transition rates. If the events in the transitions between states follow an exponential distribution, this graph is considered as a Markov Chain. The corresponding transition matrix of the Markov Chain of Figure 2 with the input rates is shown in Table I. These rates have been arbitrarily provided in order to illustrate the input parameters for determining the steady-state probabilities.

| | **{P1, W2}** | **{B1, P2}** | **{W1, P2}** | **{B1, B2}** | **{P1, B2}** |
|---|---|---|---|---|---|
| **{P1, W2}** | $\gamma_1$ (5.0) | $\beta_1$ (0.1) | 0.0 | 0.0 | 0.0 |
| **{B1, P2}** | 0.0 | $\gamma_2$ (5.0) | $\mu_1$ (3.0) | $\beta_2$(0.2) | 0.0 |
| **{W1, P2}** | $\gamma_2$(5.0) | 0.0 | 0.0 | 0.0 | $\beta_2$ (0.2) |
| **{B1, B2}** | 0.0 | $\mu_2$ (3.0) | 0.0 | 0.0 | $\mu_1$ (3.0) |
| **{P1, B2}** | $\mu_2$ (3.0) | 0.0 | 0.0 | $\beta_1$ (0.1) | $\gamma_1$ (5.0) |

Table I. Input parameters for Figure 1

This transition matrix (or Markov Chain) when solved, steady-state probabilities are obtained and shown in Table II. The solution method implemented is SOR (Successive Over Relaxation). These probabilities represent percentage of time occupied in each configuration. With these probabilities, one can easily find the steady-state probability of each basic state. For instance, in order to determine the steady-state probability of state P1, the sum of steady-state probabilities of those configurations consisting of state P1 has to be performed. In Table II, by adding 0.948475 (referring to configuration {P1,W2}) with 0.00216554 (referring to configuration {P1,B2}), the steady-state probability of state P1 which is 0.95064054 is obtained. As explained elsewhere, these are the most basic measures. Other measures such as clients in the system, throughput and others can be calculated and this is shown in Section 3.

| **Configurations** | **Steady-State Probabilities** |
|---|---|
| {P1, W2} | 0.948475 |
| {B1, P2} | 0.030631 |
| {W1, P2} | 0.0176717 |
| {B1, B2} | 0.00105715 |

| | |
|---|---|
| {P1, B2} | 0.00216554 |

Table II. Steady-state probabilities obtained for the Markov chain of Figure 2

Now that PerformCharts was presented a few words should be dedicated of how to interact with the tool. The tool was developed in C++. Therefore, the interface of the specification as well as the calls to the methods to generate the Markov chain and the calculations to perform steady-state probabilities have to be coded in the main program. In order to make this process a little more comfortable, a language PcML (PerformCharts Markup Language) based on XML (eXtensible Markup Language) was developed. This language is interpreted through a Perl script that converts it into the main program.

## 3. From Steady-State Probabilities to Performance Measures

The previous section showed how to obtain steady-state probabilities of a given system modeled in Statecharts. Based on these probabilities, several interesting metrics that refer to the system behavior in a long run can be obtained. These metrics are known as performance measures.

Performance measures are quantified values that determine certain features of a given model. For example, a queue in a market, utilization of servers of a database, inspection of machines in a maintenance center, etc. are those measures. These measures play a major role in decision making such as including one more teller to attend the queue, shut off a server, hire more personnel to repair, etc. The objective of this Section is to show the implementation included within PerformCharts tool to calculate the performance measures that are function of the steady-state probabilities.

Within the context of PerformCharts, when a Markov chain is obtained it contains a set of configurations (basic states of each orthogonal component) and when calculated, steady-state probabilities of each configuration are provided. Therefore, some basic measures such as mean quantities of input and output of each basic state of Statecharts are readily calculated. The formulae to calculate such quantities are: transition rate on the arc moving to the state in question * steady-state probability of that state; transition rate on the arc moving out of the state in question * steady-state probability of that state.

As an example, in order to determine the mean quantities of output for P1, one has to do the following calculation: (0.9484*5.0 + 0.9484*0.1) + (0.0021*3.0 + 0.0021*0.1+0.0021*5.0) = 4.8423. Table III shows these two measures provided for all the basic states of Figure 1.

| States | Mean Values of Input | Mean Values of Output |
|---|---|---|
| W1 | 0.0530197 | 0.0919009 |
| P1 | 12.3478 | 4.85479 |
| B1 | 0.24844 | 0.257532 |
| W2 | 12.3299 | 4.83712 |
| P2 | 0.301142 | 0.343087 |
| B2 | 0.0182106 | 0.0240212 |

Table III. Performance Measures for the example shown in Figure 1

Several other kinds of performance measures can also be obtained by means of steady-state probabilities, especially in systems modeled as queues. A case study in the following Section deals with one such example.

## 4. Case Study

In order to illustrate the implementation of performance measures calculation, consider a communication system in which a message arrives with a rate of 60 messages per minute (λ) and if the sender component is busy, the message is placed in a buffer. The arrival rate has an exponential distribution. The system operates at a maximum rate 52 kbps; however, there is an

influence due to noise and this rate decreases to 40 kbps (μ). Both these rates also possess an exponential distribution. The capacity for the buffer to store the messages is 3 x $10^5$ bits. The system specified in Statecharts is shown in Figure 3.

This specification is converted into a Markov chain with the buffer size of 3. Once the Markov chain is resolved, the performance measures were obtained and were compared with another approach using simulation in order to validate the implementation. The simulation approach made use of Arena software system (http://www.arenasimulation.com) in order to obtain the results.

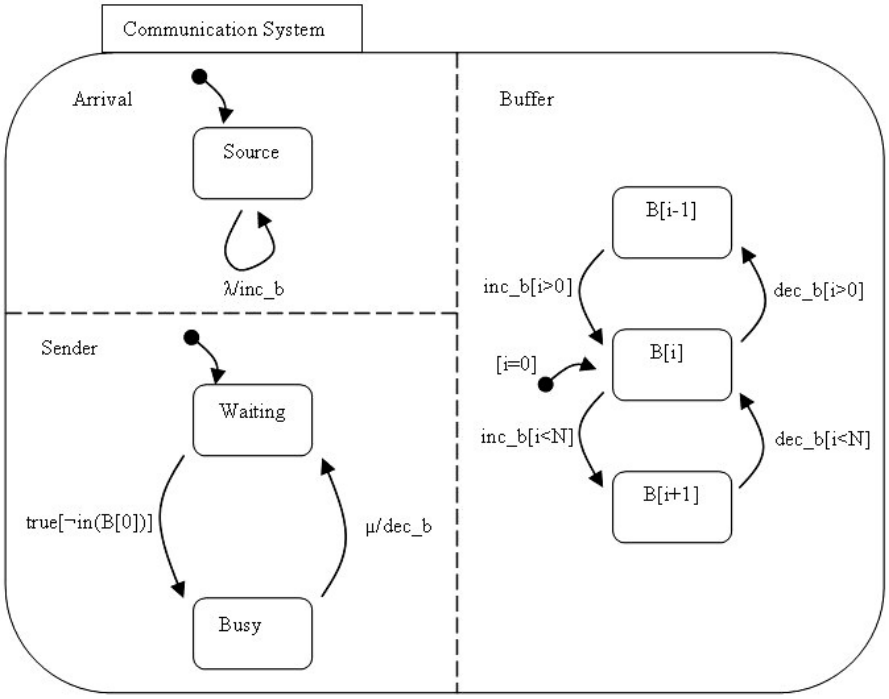


Figure 3. Communication system specified in Statecharts

Some other tests were also performed based on the same example shown in Figure 3. The buffer size was increased to 5 and 10. Again, for the buffer sizes of 3, 5 and 10 two servers, instead of one, were considered. The results when compared to those obtained from simulation were quite similar thus validating the implementation of the calculation of performance measures. Tables IV and V show the results from these runs.

| **Metrics** | **Analytical results** | | | **Simulation results** | | |
|---|---|---|---|---|---|---|
| Buffer size | 3 | 5 | 10 | 3 | 5 | 10 |
| Mean waiting time | 6.09 s | 10.89 | 23.41 | 6.06 s | 10.88 | 22.98 |
| Mean response time | 8.59 s | 13.39 | 25.91 | 8.56 s | 13.39 | 25.47 |
| Mean queue size | 2.40 | 4.34 | 9.36 | 2.39 | 4.34 | 9.29 |
| Mean number of clients in the system | 3.35 | 5.34 | 10.36 | 3.37 | 5.33 | 10.29 |

Table IV. Analytical and simulation results with one server

| Metrics | Analytical results | | | Simulation results | | |
|---|---|---|---|---|---|---|
| Buffer size | 3 | 5 | 10 | 3 | 5 | 10 |
| Mean waiting time | 2.07 s | 3.78 | 8.71 | 2.04 s | 3.79 | 8.77 |
| Mean response time | 4.57 s | 6.28 | 11.21 | 4.52 s | 6.29 | 11.27 |
| Mean queue size | 1.48 | 2.84 | 6.85 | 1.48 | 2.85 | 6.91 |
| Mean number of clients in the system | 3.27 | 4.73 | 8.82 | 3.27 | 4.74 | 8.88 |

Table V. Analytical and simulation results with two servers

## 5. Final Remarks

Modeling has become essential in dealing with complex systems in order to conduct several types of analysis so that when the actual system is in fact implemented or constructed, errors can be minimized if at all cannot be entirely eliminated. With respect to this issue, Statecharts can be considered a potential candidate to represent complex reactive systems. Moreover, it has been shown that a mathematical solution can indeed be associated with the representation in order to obtain, in this case, performance information. The paper showed how to deal with Markov chains when a complex reactive system is specified in Statecharts by presenting PerformCharts tool. In the beginning, only steady-state probabilities were the output of the tool. Now, some more routines have been incorporated to the tool to yield performance measures that are functions of the steady-state probabilities.

In order to test whether the implementation correctly calculates the performance measures, same models were also introduced into a simulation approach. The results were compared and were satisfactory thus validating the implementation.

The interface to deal with PerformCharts is a textual one but based on XML language. The language is interpreted through a perl script to generate the main program in C++. The graphical interface is under development and this interface will be converted into the PcML specification.

This same tool is also being used to generate test sequences to perform black box testing. The same logic applied for performance evaluation is also utilized in this case. The Statecharts representation of a software specification is converted into a Finite State Machine from which test sequence generating methods are employed and in turn these sequences are submitted to the implementation to determine a verdict whether the implementation conforms to its specification.

Another development is under progress that should enable the access to PerformCharts for both performance evaluation and test sequence generation via internet.

## References


**Drusinsky, D.** *Extended State Diagrams and Reactive Systems***.** Dr. Dobb's Journal**,** Vol. 19, n. 11, p.72-80, 1994.

**Harel, D**. *Statecharts: a visual formalism for complex systems*. Science of Computer Programming, Vol. 8, p. 231-274, 1987.

**Harel, D., Pnueli, A., Schmidt, J., Sherman, R.** *On the formal semantics of Statecharts*. **In**: Proceedings of IEEE Symposium On Logic In Computer Science, Ithaca, USA. p.54-64, 1987.

**Harel, D., Politi, M.** *Modeling Reactive Systems with Statecharts: the Statemate Approach*. McGraw-Hill, USA, 1998.

**Philippe, B., Saad, Y., Stewart, W. J.** *Numerical Methods in Markov Chain modeling*. Operations Research. Vol. 40, n. 6, p.1156-1179, 1992.

**Silva, E.A.S., Muntz, R.R.** *Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação***.** VIII Escola de Computação. UFRGS, Gramado, Brazil, 1992.

**Vijaykumar, N.L., Carvalho, S.V., Andrade, V.M.B., Abdurahiman, V.** *On embedding Memory in Markov Models specified in Statecharts*. **In:** Anais em CD-ROM do WPerformance – I Workshop em Desempenho de Sistemas de Computação e de Comunicação, Florianópolis, SC, Brazil, p.1394-1405, 2002.

**Vijaykumar, N.L., Carvalho, S.V., Francês, C.R.L., Abdurahiman, V.** *Performance Evaluation from Statecharts representation of Complex Systems: Markov Approach*. **In:** Anais em CD-ROM do WPerformance – V Workshop em Desempenho de Sistemas de Computação e de Comunicação, Campo Grande, MS, Brazil, p.183-202, 2006.

**Viswanadham, N., Narahari, Y.** *Stochastic Petri net models for performance evaluation of automated manufacturing systems*. Information and Decision Technologies, Vol. 14, p.125-142, 1988.